A Coarse-to-Fine Control Strategy for Stereo and Motion on a Mesh-Connected Computer

Lance R. Williams and P. Anandan

COINS Technical Report 86-19**
May, 1986

Abstract

Using image representations of multiple spatial scale as part of a *coarse-to-fine* control strategy has proven extremely useful in solving the correspondence problem inherent in stereo and motion. Unfortunately, execution times of existing implementations on sequential machines fall substantially short of real-time performance. The utility of such algorithms would be greatly increased if it could be shown that they can be implemented in parallel without placing unrealistic demands on resources of space and connectivity. By requiring that the control strategy take opportunistic advantage of a simple uniform *vergence* movement, the space and connectivity requirements of these hierarchical algorithms are drastically reduced. We present descriptions of two different hierarchical algorithms that are designed for a simple mesh-connected computer. The first is a mesh-computer implementation of a Marr-Poggio-Grimson stereo matcher which runs in $O(\delta \log \delta)$ time. The second is a mesh-computer implementation of a motion correspondence algorithm which uses a correlation based technique, requiring only $O(\delta^2 \log \delta)$ time.

** This report is sponsored by Allen R. Hanson and Edward M. Riseman. It will appear in CVPR Proceedings, Miami Beach, Fl., 1986. This research was supported by the Defense Advanced Research Projects Agency under grant N00014-82-K-0464, the National Science Foundation under grant # DCR-8318776, and the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office for Scientific Research under contract # F49620-86-C-0041.

1. INTRODUCTION

The major problem common to both stereopsis and motion analysis is computing a correspondence between features from the slightly different images comprising a stereo pair or motion sequence. A common technique used in algorithms which solve the correspondence problem is a *coarse-to-fine* control strategy (see figure 1).

If we "blur" the image by convolving with a Gaussian of a given standard deviation, we can restrict the spatial "rate" at which changes occur in the image intensity function. "Edges" computed from the convolution of the image intensity function are thus distributed at intervals proportional to the standard deviation of the Gaussian. Since with larger Gaussians, "edges" are distributed more sparsely, unique matches can be determined for larger changes in position, or *disparity*. By using a Gaussian with standard deviation of the same order of magnitude as the expected disparity, matching can be unambiguous. However, disparity *resolution* has been sacrificed for increased disparity *range*. If we "blur" the image to varying degrees, and use the results of matching features from more coarsely "blurred" images to guide our matching of features from less "blurred" images, we capture the essence of coarse-to-fine control.

Any algorithm employing this control strategy will contain several basic components: First, a matching primitive and a method for computing it at several spatial scales; Second, a method for matching within a single *spatial frequency tuned channel*; And last, a method for implementing the coarse-to-fine control between the channels. Beyond this basic similarity, the specific details of implementation vary widely.

The Marr-Poggio-Grimson (MPG) stereo matcher [1,2] is the definitive example of this class of algorithm. It represents only one of several possible combinations of choices for each of the above components. The matching primitive used in the MPG matcher is the signed zero crossing of the $\nabla^2 G$ convolved image. It can be computed at a variety of scales by changing the standard deviation of the underlying Gaussian. Matching within a single channel is accomplished by searching a small interval of the corresponding epipolar for a like signed zero crossing. The coarse-to-fine control strategy is accomplished through *vergence* movement. Any local area of a high or middle frequency channel with less than 70% matches is declared "out of range." Regions that are out of range are translated in the direction of the majority disparity within the spatially coincident area of the adjacent lower frequency. This brings them into a disparity range where matching is unambiguous (see [2]). Variations of the algorithm have been implemented by many people [2-6], who have shown it to be successful at solving the stereo correspondence problem in a wide variety of random-dot stereograms and natural image stereo pairs.

The control strategy used by Williams [6] is of particular interest, since we will employ a similar strategy here. In Williams' implementation, the left and right images are moved relative to each other in a single uniform movement. Matching is repeated at periodic intervals; matches within the higher frequency channel are accepted only if they roughly agree with the majority disparity within the spatially coincident area of the adjacent lower frequency. Matching within local areas of spatial frequency tuned channels is viewed as taking opportunistic advantage of a vergence movement that is controlled at a global level. The original motivation behind this control strategy was to provide a more accurate model of human stereopsis by reducing the demands on the resource of physical eye movement inherent in control at a local level [6]. It is interesting that this desire for simplicity, motivated by physical constraints, translates to a strategy that reduces the demands of space and connectivity required in a parallel architecture.

Coarse-to-fine control strategies have been less often used for displacement field computation in motion analysis[7,8]. An example of where it has been used is the hierarchical correlation algorithm of Glazer, *et al.* which uses local correlation of band-pass filtered images. A range of spatial frequency channels results from convolving the input images with $\nabla^2 G$ operators of increasing standard deviation and simultaneously decreasing spatial resolution. In this way, the spatial frequency channels form different levels in a pyramid data structure. The match measure between two pixels is computed through cross correlation of the local neighborhoods surrounding each pixel. For each pixel in the first frame, the corresponding pixel in the second frame is the one for which the match measure is optimized. The coarse-to-fine control strategy operates by using the displacement estimate of a pixel at a coarse level (lower frequency channel) as the initial estimate for all daughter pixels in the pyramid at the next finer level (higher frequency channel).

Existing implementations of correspondence algorithms using coarse-to-fine control run at speeds substantially short of real-time [4]. Nishihara's PRISM system is a hierarchical stereo matcher that was designed to provide a substantial increase in execution speed. Although the system exploited parallelism only for computing the required $\nabla^2 G$ convolutions, impressive speed gains were reported [4]. Unfortunately, the system still falls short of what would be called real-time performance, requiring about 30 seconds to process a typical stereo pair. Clearly, a higher degree of parallelism will have to be employed before this class of algorithm sees routine use in robotics.

Within the next few years, a new generation of fine-grained, massively parallel machines will come into operation. These include the Connection Machine at MIT [9], NON-VON at Columbia University [10], and the CAAPP at the University of Massachusetts, [11]. A conservative configuration for each of these machines resembles what is often termed a *mesh-connected computer*, or MCC. We adopt the definition of MCC provided by Miller and Stout [12]. The MCC is a single instruction stream-multiple data stream (SIMD) computer. It is composed of an array of processors arranged in an $n \times n$ matrix (see figure 2). Each processor has a unique identification number representing the address of that processor in row-major form. From this number, the absolute x and y address of the processor can be computed. Each processor has a constant number of registers of word size $\theta(\log n)$ for a total of $O(\log n)$ space. Thus, any register can hold the absolute address of any processor in the matrix. Each processor can ship a single word of data to its east, west, north, or south neighbor in $\theta(1)$ time and can perform standard arithmetic on the contents of its registers in $\theta(1)$ time.

The highly parallel character of coarse-to-fine correspondence algorithms has long been recognized. In fact, the motion correspondence algorithm we present was developed in the environment of the University of Massachusetts' VISIONS system [7]. The VISIONS system simulates a hierarchy of mesh-connected computers of decreasing spatial resolution connected in pyramid or quad-tree fashion [13]. However, to the authors' knowledge, there has been no complete specification of any similar algorithm for the relatively simple mesh-connected computer. Since, unlike many other abstract parallel machines, the mesh-connected computer seems to lie well within the scope of what is practically achievable in current semiconductor technology, we feel the specification of coarse-to-fine correspondence algorithms for this class of machine to be an important step towards the realization of a real-time vision system.

2. PARALLEL LANGUAGE CONVENTIONS

The MCC algorithms we present in this paper are written in a hybrid notation which borrows features from N-PASCAL (a language for expressing SIMD algorithms for the NON-VON [14]). The principal features of N-PASCAL of interest to us here are the *vector* variable type and the parallel conditional form, *where-do-elsewhere*. Statements in N-PASCAL containing references to vector variables operate in parallel on all processing elements. The *where-do-elsewhere* form allows the execution of a block of code on a processing element to be conditional on the value of a vector variable. When a *where* is encountered, execution on all processing elements which satisfy the vector variable condition proceeds while execution on all other processing elements is temporarily suspended. An optional *elsewhere* allows a block of code to be executed by those processing elements which failed the original condition.

The convention we use for vector variables, or registers, is a variable name in italic capital letters optionally followed by a compass direction in parenthesis, (e.g., A(east)). The legal register transfers permitted within the constraints imposed by our definition of MCC are thus assignments to A from A(east), A(west),A(north) and A(south).

3. STEREOPSIS ALGORITHM

The algorithm we present for stereopsis is relatively simple, and consists of three main routines: CONVOLVE, VERGE and FIND_MAJORITY. CONVOLVE computes the $\nabla^2 G$ convolution of the left and right images. VERGE first finds the zero crossings in the output of CONVOLVE. It then translates the zero crossing representation of the right image relative to the left along the x-axis, noting when like-signed zero crossings become "aligned." It records those disparities which satisfy a *spectral continuity* constraint [6]. FIND_MAJORITY computes the majority disparity within all local neighborhoods for use by VERGE in its spectral continuity check. CONVOLVE and VERGE are repeated, in sequence, for each of $\log \delta$ spatial frequencies to be processed. Since the lowest frequency requires no spectral continuity check, FIND_MAJORITY need only be executed $(\log \delta - 1)$ times. The *level numbers* increase from 0 to $\log \delta$ as the frequencies decrease from high to low. We refer to the level number of a specific channel as l. For space considerations, the code we present is intended to be more illustrative than exhaustive. When code is omitted, we note it with a remark.

CONVOLVE

CONVOLVE assumes that the left image is in register, I. Since $G_{2d}(x,y) = G_{1d}(x) \cdot G_{1d}(y)$, where G_{2d} is the two-dimensional and G_{1d} is the one-dimensional Gaussian mask, the $\nabla^2 G$ convolution mask can be expressed in the following form:

$$\nabla^2 G_{2d}(\sigma, x, y) = G_{1d}(\sigma, y) \frac{\partial^2 G_{1d}(\sigma, x)}{\partial x^2} + \frac{\partial^2 G_{1d}(\sigma, y)}{\partial y^2} G_{1d}(\sigma, x)$$

The $\nabla^2 G$ convolution can be expressed as the sum of the two separable convolutions corresponding to the terms in the above expression. A similar analysis can be found in [15].

Each processor in the mesh first multiplies the contents of its I register by the value of the $\frac{\partial^2 G_{1d}}{\partial x^2}$ function at the maximum mask radius, r (see figure 3). This becomes the initial value of a partial sum that is then passed to its west neighbor. Each processor then multiplies the contents of its I register by the function evaluated at $r - 1$. It is added to the partial sum it *received* from its east neighbor. The new sum is now passed to its west neighbor and the process continues, forming the partial sum as the value is propagated to the west. A mirror image process simultaneously computes a right partial sum, which, when added to the sum from the left, constitutes the one dimensional convolution for a row. The final two dimensional convolution is computed by convolving the output of the row convolution in a similar fashion, except along columns, with the Gaussian function. This takes advantage of the separable filter properties of $\nabla^2 G$ and the Gaussian. The other term is computed in a similar fashion and the final output is the sum of the two terms. The right image is convolved similarly.

```
procedure CONVOLVE($l$ : integer);

begin

$\sigma := 2^l$; $r := \rho * 2^l$;

/* do Laplacian convolution in x-direction */
for i := 0 to $(r-1)$ do begin
   /* partial sums for east and west halves of Laplacian */
   $EPS := EPS(\text{east}) + I * \frac{\partial^2 G_{1d}}{\partial x^2}(\sigma, r-i)$;
   $WPS := WPS(\text{west}) + I * \frac{\partial^2 G_{1d}}{\partial x^2}(\sigma, r-i)$;
   end;

/* add east and west sums */
$LAP := EPS(\text{east}) + WPS(\text{west}) + I * \frac{\partial^2 G_{1d}}{\partial x^2}(\sigma, 0)$;

/* do Gaussian convolution in y-direction */
for i := 0 to $(r-1)$ do begin
   /* partial sums for north and south halves of Gaussian */
   $NPS := NPS(\text{north}) + LAP * G_{1d}(\sigma, r-i)$;
   $SPS := SPS(\text{south}) + LAP * G_{1d}(\sigma, r-i)$;
   end;

/* add north and south sums */
$GAUSS := NPS(\text{north}) + SPS(\text{south}) + LAP * G_{1d}(\sigma, 0)$;

   ...
/* we omit similar code which computes the second term */
   ...

end;
```

VERGE

VERGE assumes that $\nabla^2 G$ convolved left and right images reside in registers L and R and checks for zero crossings by comparing the signs of adjacent values. Zero crossings are stored in the LZ and RZ registers, $+1$ for positive signed crossings, -1 for negative signed crossings, and 0 otherwise. The *vergence* movement, or translation, is accomplished by repeatedly passing the contents of the RZ register to the east neighbor. We assume that the original alignment of the images is such that all matches will be found during the course of this simple translation; that is to say, all matches are of *convergent* sign[1]. The contents of the LZ and RZ registers are compared at every point along the translation, the extent of which equals the disparity range being searched. When the LZ and RZ registers are equal, the disparity is compared to the majority disparity within the spatially coincident area of the adjacent lower frequency channel, found in the MAJ register. If the difference in disparity is less than 2^l, the *match* is accepted, otherwise it is rejected. This explicit *spectral continuity* check is required in all but the lowest spatial frequency tuned channel, where matches are assumed to be unambiguous over the disparity range being searched.

```
procedure VERGE($\delta$,$l$ : integer);

begin

LZ := RZ := 0;
/* find signed zero crossings */
where ($L \geq 0$) and ($L$(east) $< 0$) do $LZ := 1$;
where ($L \leq 0$) and ($L$(east) $> 0$) do $LZ := -1$;
where ($R \geq 0$) and ($R$(east) $< 0$) do $RZ := 1$;
where ($R \leq 0$) and ($R$(east) $> 0$) do $RZ := -1$;

$MAP := \infty$;

/* translate right image, checking correspondence */
for i := 0 to $\delta$ do begin

  /* if aligned and spectrally continuous, save disparity */
  where ($LZ = RZ$) and ($LZ \neq 0$) and ($|i - MAJ| \leq 2^l$) do
    $MAP := i$;
  /* translate right image */
  RZ := RZ(west);
  end;
end;
```

FIND_MAJORITY

FIND_MAJORITY assumes that a disparity map computed by VERGE lies in the *MAP* register. It begins by considering support for the first disparity. If a processor's *MAP* register contains a disparity within 2^l of the disparity under consideration, then the *FLAG* register is assigned the value 1, otherwise it is assigned the value 0. FIND_MAJORITY then counts the number of non-zero *FLAG* registers by "convolving" with a uniform unit mask and accumulating the partial sums in a manner identical to that used in CONVOLVE. The resulting count is placed in the *COUNT* register. It is also possible to compute support for the current disparity by convolving with a Gaussian, instead of the uniform unit mask. Support for the disparity decreases with increasing distance, weighted by the Gaussian distribution, in accordance with some recent work of Prazdny [16]. The convolution output is then compared to the current maximum (zero in this case) which is stored in *MAX*. The entire process is repeated for the next disparity, which is computed by adding 2^l to the previous candidate. The disparity range is thus sub-sampled at rate 2^l. After the last disparity is considered, the *MAJ* register contains the disparity with maximum support.

```
procedure FIND_MAJORITY($\delta$,$l$ : integer);

begin

$MAX$ := $MAJ$ := 0;
/* determine support for each disparity */
for i := 0 to $\delta$ step $2^l$ do begin
  where $| MAP - i | \leq 2^l$ do $FLAG$ := 1 elsewhere do $FLAG$ := 0;

    ...
  /* count the number of flagged disparity values */
    ...
  /* compare support for current disparity to maximum */
  where $COUNT > MAX$ do begin
    $MAX := COUNT$;
    $MAJ := i$;
    end
  end
end;
```

COMPLEXITY ANALYSIS

If δ is the maximum disparity considered, then the CONVOLVE process requires $O(\delta)$ time for the lowest spatial frequency. Total time required for convolution of $\log \delta$ channels is thus no more than $O(\delta \log \delta)$. VERGE requires $O(\delta)$ time for each of $\log \delta$ spatial frequency channels. Thus total time required for vergence movement is also $O(\delta \log \delta)$. Examination of FIND_MAJORITY reveals that for any spatial frequency channel, the number of disparity candidates considered is $\delta/2^l$. Computing support for each candidate requires $O(2^l)$ time. Thus for a single channel, execution time is $O(\delta)$. Since FIND_MAJORITY is executed once for each of $(\log \delta - 1)$ channels, total execution time is $O(\delta \log \delta)$. The mesh-computer stereopsis algorithm we present thus requires only $O(\delta \log \delta)$ time. We note that the hypothetical lower bound for a stereopsis algorithm using a coarse-to-fine control strategy is $O(n^2)$ on a sequential machine [3], where $n \times n$ is the image size. The execution time is thus a function of image size, n, whereas on the MCC, it is a function of disparity, δ. Although in practice, disparity and image size are often related, usually disparity is only a small fraction of the image size. In such cases, a significant advantage exists on the MCC.

4. MOTION ALGORITHM

This section contains a description of a motion algorithm for the MCC which is based on the hierarchical correlation matching algorithm presented in [7]. Although some recent improvements have been made to this algorithm (see [17,18]), we restrict our discussion to the original, due to its simplicity. The improvements are also easy to implement on an MCC, but since our intention is primarily to illustrate the control strategy, we do not include them.

As mentioned earlier, Glazer's algorithm was designed for a pyramid architecture, while we are restricted to the simpler MCC. We make four modifications to convert this algorithm to the MCC. First, within each channel, our search strategy is a two-dimensional version of the vergence strategy we used in our stereopsis algorithm. This differs from Glazer's strategy which assumes immediate access of information at distant pixels, which is at best difficult and perhaps impossible to implement on an MCC due to connectivity constraints. Second, Glazer's algorithm assumes that the resolution of the image decreases by a factor of 2 between successive levels. This is difficult to implement on the MCC; instead, all levels are represented at the full resolution of the original images. Third, the decrease in resolution allows Glazer to search for a match within a 3×3 pixel area at all levels of the hierarchy. On the MCC, due to the full resolution representation, the length of the search area will increase by a factor of 2 with each successive coarser level. However, as we describe later, only a set of 3×3 displacements within this area are considered as candidates. Finally, maintaining full resolution images requires that the template window sizes used for computing correlation increase by a factor of 2 between successive levels, whereas in Glazer's strategy this remains constant (but still covers the same image area).

Our algorithm consists of three major components: CONVOLVE, SPIRAL, and CORRELATE. CONVOLVE computes the band-pass filtered images using $\nabla^2 G$ convolution. SPIRAL translates one image relative to the other through all displacements within the expected maximum displacement radius. CORRELATE estimates the similarity between the convolution values in the two overlaying windows around a pixel. The current best match location as well as its corresponding match measure are maintained.

The procedures CONVOLVE and SPIRAL are called for each level from the coarsest level ($l = \log \delta$) to the finest level ($l = 0$). Note after the algorithm terminates, each pixel in image *I1* contains the address of the matching pixel in *I2*, rather than the displacement. From this, the displacement can be easily computed.

SPIRAL

At the heart of this algorithm is the process SPIRAL. This process translates one image relative to the other through all displacements of interest. If δ is the maximum displacement in each coordinate direction, then there are $(2\delta + 1)^2$ possible relative positions at the finest level of the hierarchy. These positions can be traversed by a "spiral movement" of one image relative to the other. We illustrate this movement in figure 5. At each subsequent coarse level, the number of relative positions considered decreases by a factor of 4. Thus, although we do not reduce the resolution (the way Glazer does), we sub-sample the possible displacements to reduce the number of candidates considered. Figures 5a, 5b, 5c illustrate the relative movement at three successive levels. Note that although the number of displacements considered decreases by a factor of 4 at every subsequent level, the time taken for traversal along the spiral paths decreases only by a factor of 2.

Registers *F1* and *F2* contain the results of the $\nabla^2 G$ convolution with the two input images. The image in register *F1* is always held in registration with the processor array, whereas the other image is moved relative to it. Register *ID* contains the address of each processor, and *ID_X* and *ID_Y* are used to store the x and y coordinates of the address. For each pixel of the image in *F1*, registers *CUR_DX* and *CUR_DY* contain the location of the matching pixel in image *F2*, at the current level of the hierarchy. These are also used during the CORRELATE process to maintain the current best match location. At the beginning of the spiral movement at each level, these match locations actually correspond to the results from the adjacent coarser level. Hence, they are moved to registers *COARSE_DX* and *COARSE_DY*.

```
procedure SPIRAL($l$ : integer /* $l$ is the level number */);

begin
num_loops := $\delta/2^l$

/* initialize registers */
COARSE_DX := CUR_DX;
COARSE_DY := CUR_DY;
```

```
$ID\_X$ := $ID$ mod n; $ID\_Y$ := $ID$ diff n;

/* each loop cycle below corresponds to one spiral cycle.
   radius contains the radius of the current cycle */
/* at each step we move over $2^l$ pixels, thus sampling the
   displacements. At the finest level each pixel is traversed,
   whereas at coarse levels they get sparser */

CORRELATE($l$);

for radius := 1 to num_loops do begin

   /* move left over $2^l$ pixels once */
   for i := 1 to $2^l$ do begin
      F2 := F2(east); $ID\_X$ := $ID\_X$(east); $ID\_Y$ := $ID\_Y$(east);
      end;
   CORRELATE($l$);

   /* move up over $2^l$ pixels $2 * radius - 1$ times */
   /* this implements the arm of the spiral moving north */
   for j := 1 to $2 * radius - 1$ do begin
      for i := 1 to $2^l$ do begin
         F2 := F2(south);
         $ID\_X$ := $ID\_X$(south); $ID\_Y$ := $ID\_Y$(south);
         end;
      CORRELATE($l$);
      end;

      ...
   /* move east over $2^l$ pixels $2 * radius$ times */
   /* this implements the arm of the spiral moving east */
      ...
   /* move south over $2^l$ pixels $2 * radius$ times */
   /* this implements the arm of the spiral moving south */
      ...
   /* move west over $2^l$ pixels $2 * radius$ times */
   /* this implements the arm of the spiral moving west */
      ...

   end;
end;
```

CORRELATE

CORRELATE computes the correlation measure of the values in the two overlaying windows. At any pixel, this measure is computed only for those displacements that are within a certain radius from the displacement estimate from the coarse level. This radius corresponds to half the search radius at the previous coarser level. When the relative position of the images is such that the displacement exceeds this radius, the processor for that pixel is shut down. Due to the fact that we sub-sample the potential displacements, at all levels there will only be 3×3 displacements that are considered for each pixel (see 5). The process of computing the correlation measure involves multiplying the convolution values from corresponding pixels of the two images and summing the resulting products within a window around each pixel. The summing process is similar to CONVOLVE, except that the weights are uniform for all pixels. The correlation window size decreases as the processing moves from coarse to fine levels. CORRELATE determines the match for each pixel by maintaining a running best match estimate and comparing the correlation measure for the current displacement with that of the current best match estimate.

In addition to the registers specified for SPIRAL, the register *CUR_MAX* contains the current maximum correlation measure.

```
procedure CORRELATE(level : integer);

begin

/* determine the distance of the current displacement
  from the coarse estimate */
DIST_X := |COARSE_DX - ID_X|;
DIST_Y := |COARSE_DY - ID_Y|;

/* multiply corresponding values */
C := F1 * F2;

  ...
/* we omit the code for summing the products */
  ...

/* running maximum selection */
where ($DIST\_X \leq 2^l$) and ($DIST\_Y \leq 2^l$) do
```

```
  where ($C > CUR\_MAX$) do begin
    $CUR\_MAX := C$;
    $CUR\_DX := ID\_X$; $CUR\_DY := ID\_Y$;
    end;
end;
```

COMPLEXITY ANALYSIS

Let δ be the maximum displacement along either coordinate direction. The convolution process takes $O(\delta)$ time at the coarsest level. Traversing over all the possible relative positions takes $O(\delta^2)$ time at the finest level. However, due to the sparser sampling of the displacements, this reduces by a factor of two at each subsequent level. Hence, at level l, the traversal time is $O(\frac{\delta^2}{2^l})$. At level l the radius of the correlation window is $2 \cdot 2^l$, hence, the time taken for the correlation summing is $O(2^l)$. Combining the traversal time with the summing time, the total time for any level l can determined to be $O(\delta^2)$. Therefore, the total time required for the motion algorithm over the $\log \delta$ levels is $O(\delta^2 \log \delta)$. If the hierarchical processing strategy is not used and the process is restricted to a single level (i.e., the image resolution), then the time required is $O(\delta^3)$.

We also note that we have used a fixed number of registers in our algorithm, none whose length exceeds $\log n$. Hence, we are within the space constraints imposed by our definition of the MCC.

5. EXPERIMENTAL RESULTS

These algorithms have been simulated within the University of Massachusetts' VISIONS system. The input images for our stereo simulation are shown in figures and The resolution is 128×128 pixels. The result of the stereo algorithm is shown in figure . The vectors displayed represent the majority disparity for zero crossings within a local neighborhood surrounding the point. In order to enhance visibility, only a subset of the vectors are displayed. The motion sequence used in our motion simulation is shown in figures and . The motion between the frames consists of a rotation in the image-plane about a point outside (to the left of) the image. The results of the displacement process are shown in figure . The resolution is 128 × 128 pixels. However, in order to enhance visibility only a sixteenth of the displacement vectors have been shown.

6. CONCLUSION

We have presented efficient algorithms for a simple mesh-connected computer which compute correspondence in stereo and motion. This was made possible by the development of a coarse-to-fine control strategy which requires that local matching processes take opportunistic advantage of a single uniform translation of the images comprising the stereo pair or motion sequence. In stereo, this translation consists of a simple "vergence" movement along the epipolar, whereas for motion, a more complex "spiral" walk is described. Computer simulations of the algorithms on a sequential machine have given promising results.

ACKNOWLEDGMENTS

Portions of this work were completed at the Computer Systems Laboratory of the Department of Computer Science at the Pennsylvania State University, University Park, PA. We wish to thank Michael Boldt, Igor Pavlin, Harpreet Sawhney and Brian Burns for their comments on our algorithms. We also thank Prof. Al Hanson for his comments on the manuscript.

References

[1] Marr, D., and T. Poggio. A computational theory of human stereo vision., *Proc. R. Soc. Lond. B204*, pp. 301-328., 1979.

[2] Grimson, W.E.L. A computer implementation of a theory of human stereo vision., *MIT AI Lab.*, Cambridge, MA, Memo 565., 1980.

[3] Grimson, W. E. L. Computational experiments with a feature based stereo algorithm, *IEEE T-PAMI*, vol. PAMI-7, pp. 17-34, 1985.

[4] Nishihara, H. K. PRISM: a practical real-time imaging stereo matcher. *MIT AI Lab.*, Cambridge, MA, Memo 780., 1984.

[5] Kak, A.C., Depth perception for robots. *School of Electrical Engineering, Purdue University, Lafayette, IN*, TR-EE-83-44., 1984.

[6] Williams, L. R., Spectral continuity and eye vergence movement, *Proc. Ninth IJCAI*, pp. 985-987, 1985.

[7] Glazer, F., Reynolds, G. and Anandan, P., Scene Matching by Hierarchical Correlation, *IEEE CVPR conference*, June 1983, pp. 432-441.

[8] Burt, P. J., Yen C. and Xu X., Multi-Resolution Flow-Through Motion Analysis, *IEEE CVPR Conference Proceedings*, June 1983, pp. 246-252.

[9] Hillis, W. D., The connection machine, *MIT AI Lab., Cambridge*, MA, Memo 646, 1981.

[10] Shaw, D. E., The NON-VON supercomputer, *Dept. of Computer Science, Columbia University Technical Report*, 1982.

[11] Weems, C. C., Image processing on a content addressable array parallel processor., *COINS Technical Report 84-14*. Dept. of Comp. and Info. Science, University of Massachusetts,

Amherst, Ma., 1984.

[12] Miller, R. and Q. F. Stout, Geometric algorithms for digitized pictures on a mesh-connected computer. *IEEE T-PAMI,* vol. PAMI-7, pp. 216-228, 1985.

[13] Hanson, A. R. and E. M. Riseman, Processing cones: a computational structure for image analysis, in *Structured computer vision*, S. Tanimoto and A. Klinger (eds.), Academic Press, 1980.

[14] Ibrahim, H. A. H., Kender, J. R., and Shaw, D. E., The Analysis and Performance of Two Middle-Level Vision Tasks on a Fine-Grained SIMD Tree Machine, *Proc. CVPR conference*, pp. 248-256, 1985.

[15] Huertas, A. and Medioni, G. Edge Detection with Subpixel Precision, *Proc. of the third workshop on computer vision*, pp. 63-74, 1985.

[16] Prazdny, K., Detection of binocular disparities, *Biological Cybernetics*, 52, pp. 93-99, 1985.

[17] Anandan P., Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion, *SPIE Intelligent Robots and Computer Vision Conference*, Vol. 521, pp. 184-194, 1984, also *COINS Technical Report 84-32*, University of Massachusetts, December 1984.

[18] Anandan, P. and Weiss, R., Introducing a Smoothness Constraint in a Matching Approach for the Computation of Displacement Fields, *DARPA IU Workshop Proc.*, pp. 186-196, 1985.

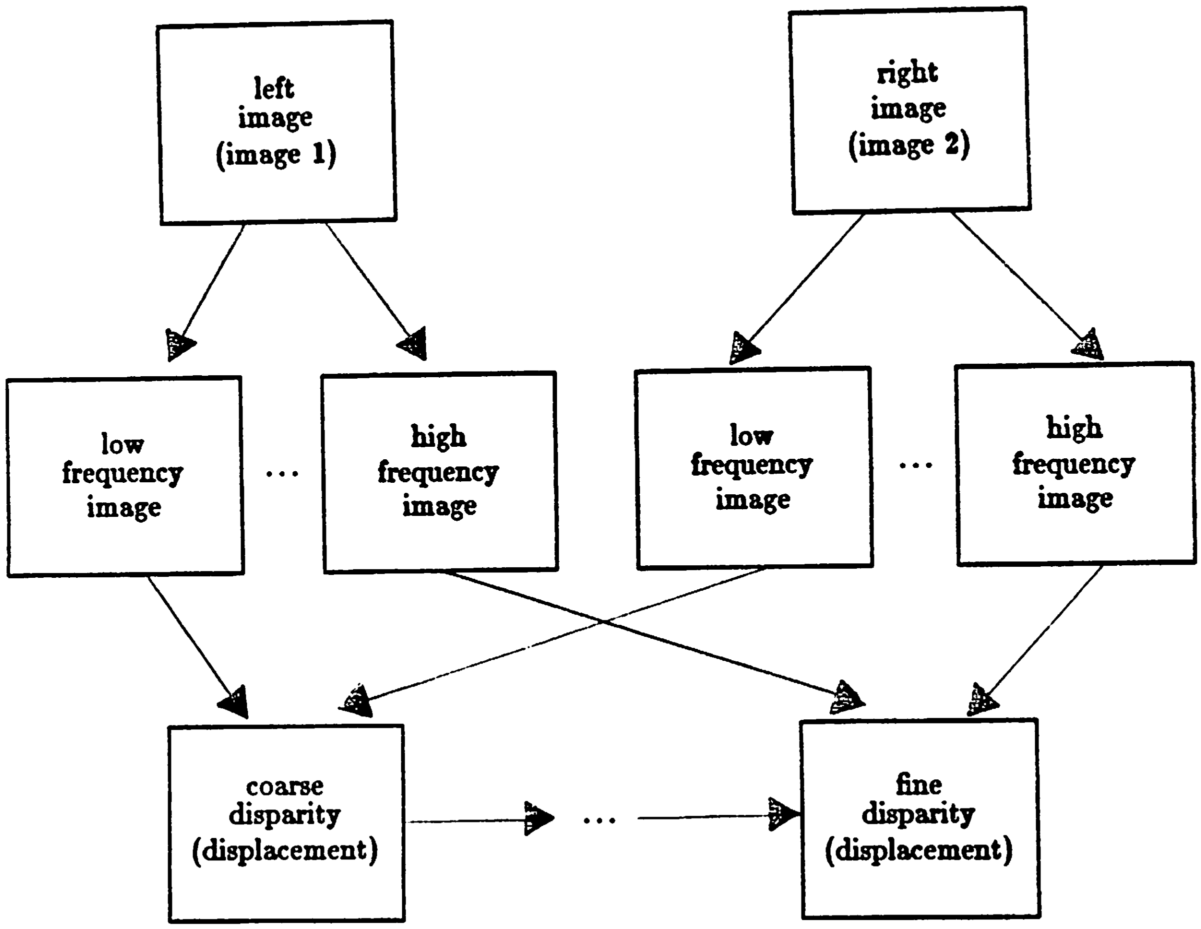


Figure 1: The information flow in our algorithms. This figure applies for both stereo and motion.

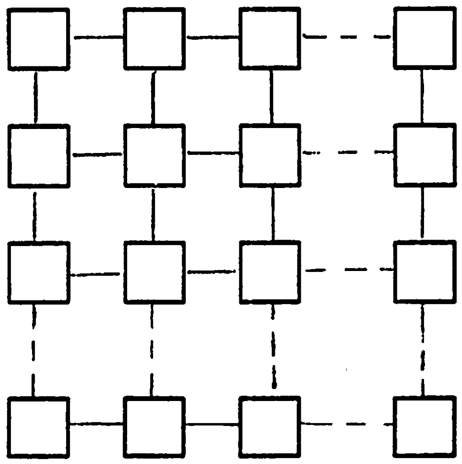
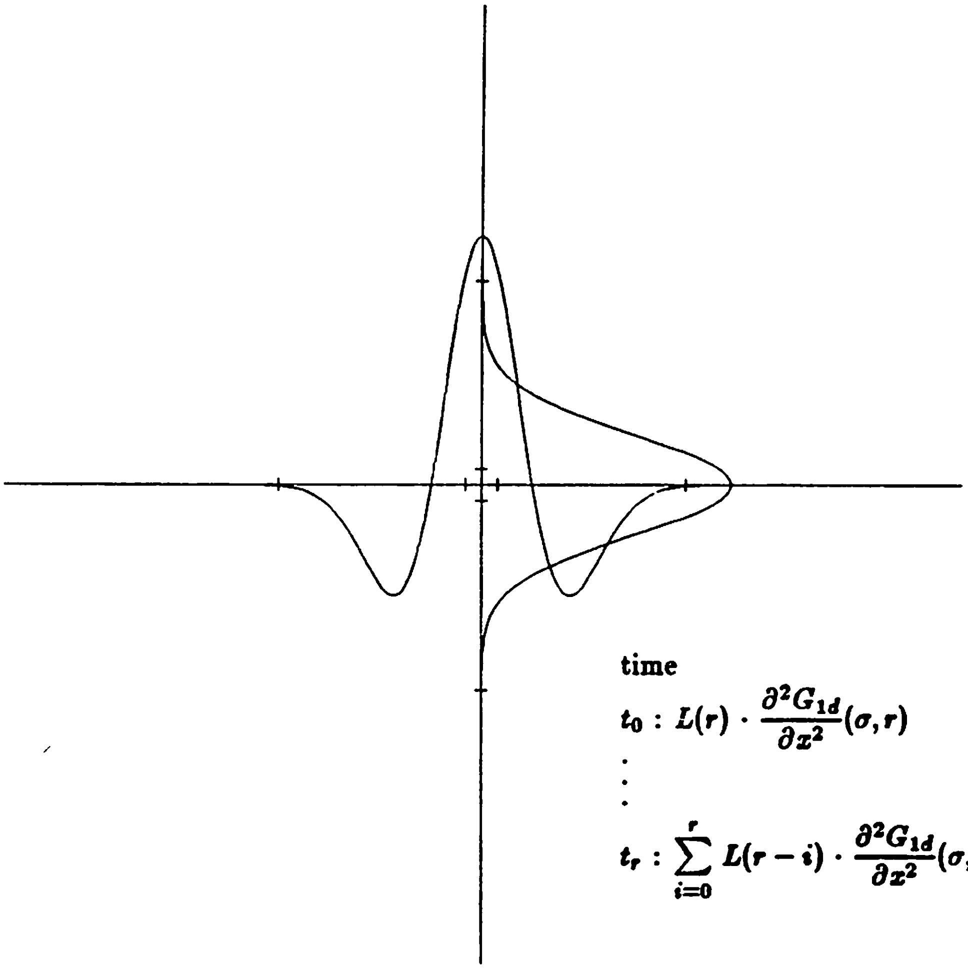


Figure 2: The organization of the PEs in the MCC.



$$t_0 : L(r) \cdot \frac{\partial^2 G_{1d}}{\partial x^2}(\sigma, r)$$

$$\vdots$$

$$t_r : \sum_{i=0}^{r} L(r-i) \cdot \frac{\partial^2 G_{1d}}{\partial x^2}(\sigma, r-i)$$

Figure 3: The two one dimensional masks used for computing one half of the $\nabla^2 G$ convolution. ($\sigma = 3.2$). The values outside a radius $r = 4\sigma$ are ignored. The summing process for the horizontal right portion is also shown.

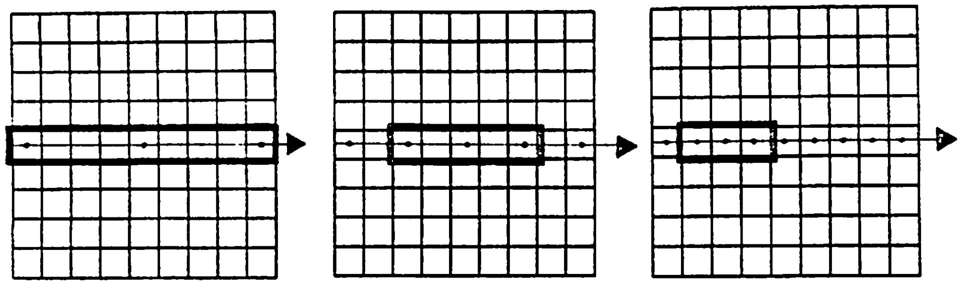


Figure 4: The vergence strategy for the stereopsis algorithm. The movement is shown for three successive spatial frequencies. The points marked are the disparities considered by FIND_MAJORITY. The dark boxes represent the refinement of the search areas which follows from the use of the spectral continuity constraint for a single processor.

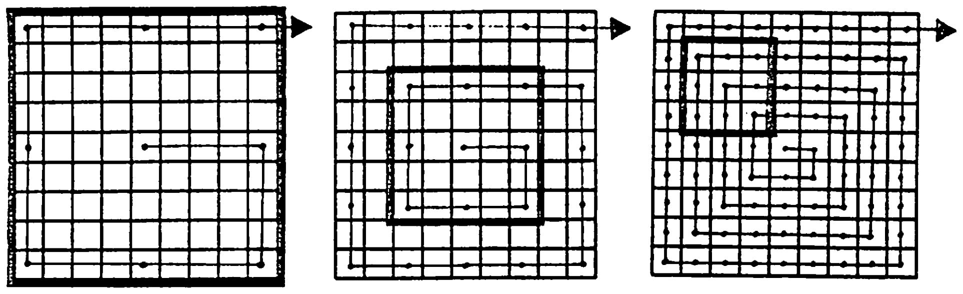


Figure 5: The spiral movement used by the motion algorithm. The nodes marked are the ones at which the traversal stops for the computation of the correlation measure. Note that the number of nodes increases by a factor of 4 at each successuve finer level, while the path length increases by a factor of 2. The dark boxes represent the refinement of the search area for a single processor.

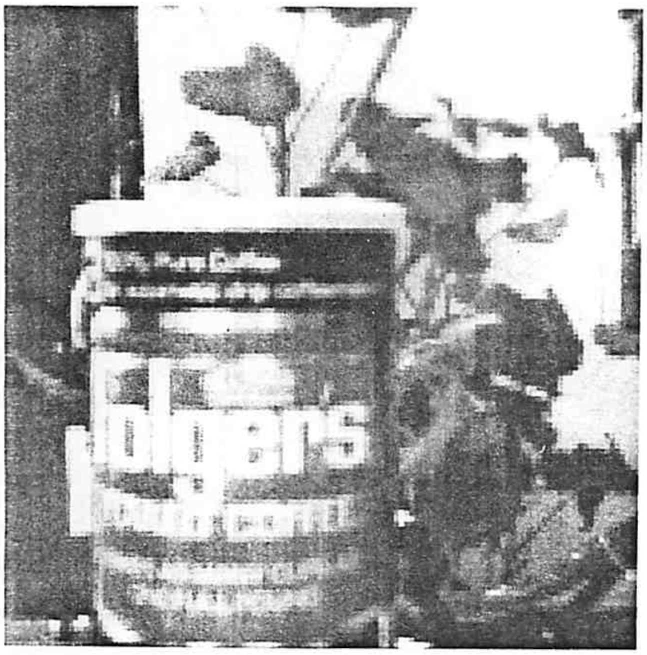


Figure 6: The left image of the stereo pair

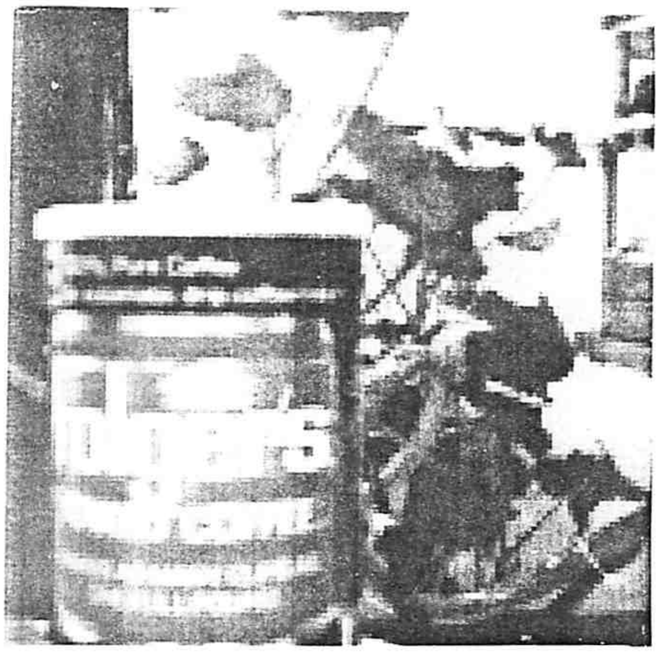


Figure 7: The right image of the stereo pair

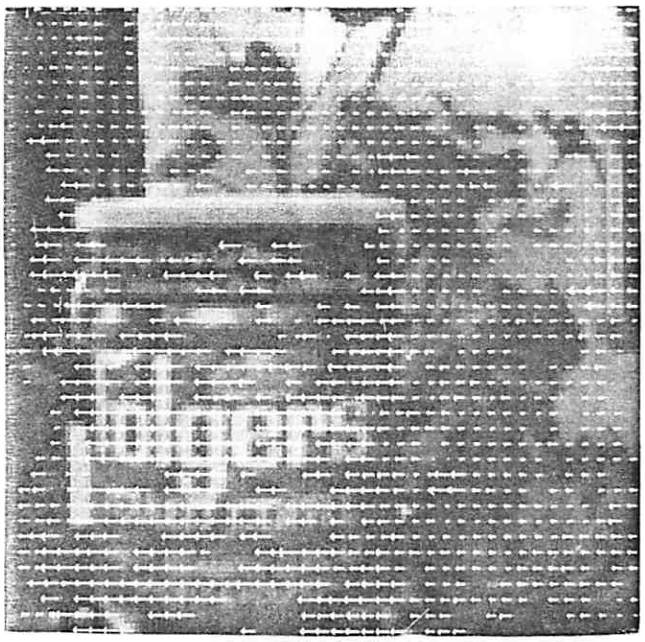


Figure 8: The disparity map for the folgers stereo pair.

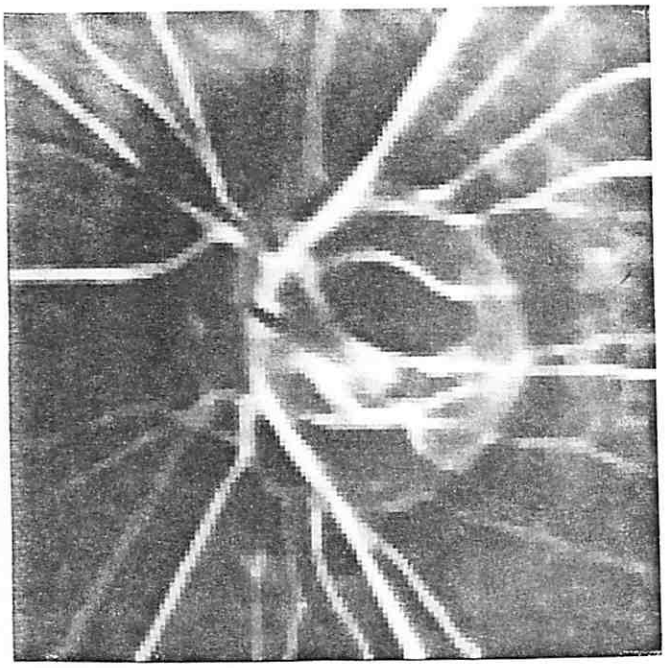


Figure 9: The first image of the optic fundus motion image sequence.

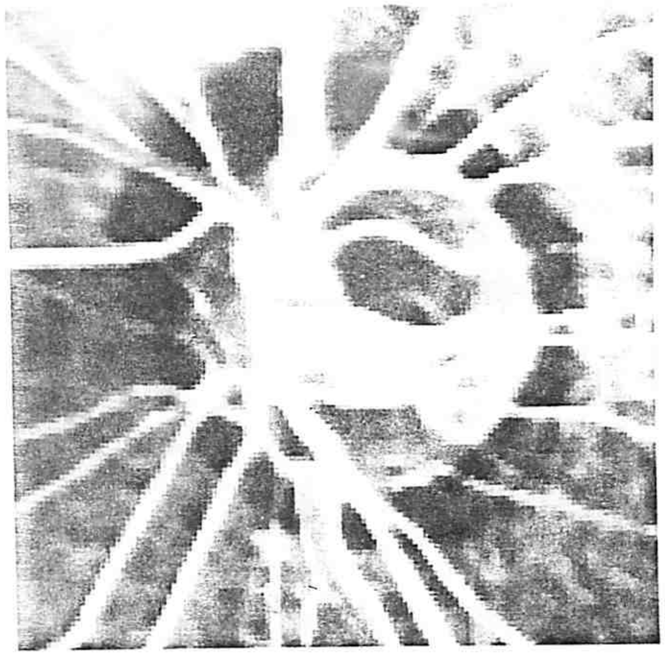


Figure 10: The second image of the optic fundus motion image sequence.

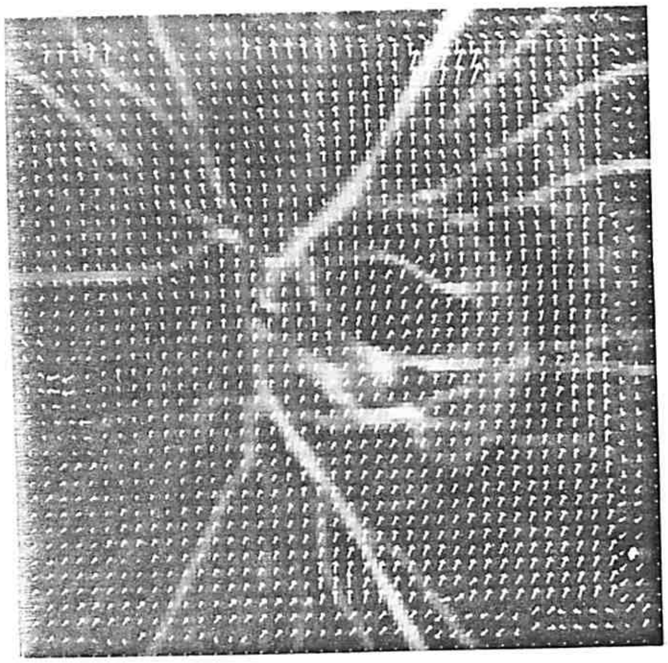


Figure 11: The displacement field for the optic fundus image sequence.